# .NET Glossary

| | |
|---|---|
| **Access control list (ACL)** | A list that specifies the rules for access to a particular resource. Microsoft Windows NT implements user-based ACLs, which specify the resources available to a particular user. |
| **Application domain (AppDomain)** | A boundary that the common language runtime establishes around objects created within the same application scope (in other words, anywhere along the sequence of object activations beginning with the application entry point). Application domains help isolate objects created in one application from those created in other applications so that run-time behavior is predictable. Multiple application domains can exist in a single process. |
| **Application domain security policy** | The security policy that a host sets on an application domain. See also: security policy. |
| **Ado** | The suite of data access technologies included in the .NET Framework class libraries. |
| **ASP.NET** | The framework for building server-based Web applications. An evolution of ASP into the managed space. |
| **ASP.NET page** | A component of an ASP.NET application. |
| **ASP.NET server control** | A server-side component that encapsulates user interface and related functionality. An ASP.NET server control derives directly or indirectly from **System.Web.UI.Control**. ASP.NET server controls comprise a superset that includes Web server controls, HTML server controls, mobile controls, and so on. An ASP.NET server control has a **runat=server** attribute on a page. See also: Web server control, HTML server control. |
| **Assembly** | A collection of functionality built, versioned, and deployed as a single implementation unit (one or multiple files). An assembly is the primary building block of a .NET application. All managed types and resources are marked either as accessible only within their implementation unit or as exported for use by code outside that unit. In the runtime, the assembly establishes the name scope for resolving requests and the visibility boundaries are enforced. The runtime can determine and locate the assembly for any running object because every type is loaded |

©2001 Microsoft Corporation

| | |
|---|---|
| | in the context of an assembly.<br><br>An assembly is the primary building block of a .NET Framework application. It is a collection of functionality that is built, versioned, and deployed as a single implementation unit (as one or more files). All managed types and resources are marked either as accessible only within their implementation unit, or as accessible by code outside that unit.<br><br>Assemblies are self-describing by means of their manifest, which is an integral part of every assembly. |
| **Assembly cache** | A machine-wide code cache used for side-by-side storage of assemblies. There are two parts to the cache: the global assembly cache contains assemblies that are explicitly installed to be shared among many applications on the computer; the download cache stores code downloaded from Internet or intranet sites, isolated to the application that triggered the download so that code downloaded on behalf of one application/page does not impact other applications. See also: global assembly cache |
| **Assembly manifest** | An integral part of every assembly that renders the assembly self-describing. The manifest establishes the assembly identity, specifies the files that make up the assembly implementation, specifies the types and resources that make up the assembly, itemizes the compile-time dependencies on other assemblies, and specifies the set of permissions required for the assembly to run properly. This information is used at run time to resolve references, enforce version-binding policy, and validate the integrity of loaded assemblies. The self-describing nature of assemblies also helps makes zero-impact install and XCOPY deployment feasible. |
| **Assertion (or Assert)** | In .NET Framework security, ensures that a method has access to a particular resource even if the method's callers do not have the required permission. During a stack walk, if a stack frame asserting the required permission is encountered, a security check for that permission will succeed. Assertions can create security holes and should only be used with extreme caution. |
| **Attribute** | A descriptive declaration that annotates programming elements such as types, fields, methods, and properties. Attributes are saved with the metadata of a .NET Framework file and can be |


©2001 Microsoft Corporation

| | |
|---|---|
| | used to describe code to the runtime or to affect application behavior at run time. |
| **Authentication** | In .NET Framework security, the process of discovering and verifying the identity of a principal by examining the user's credentials against some authority. See also: principle. |
| **Authorization** | In .NET Framework security, the process of determining whether a principal is allowed to perform a requested action. Authorization occurs after authentication and uses information about the principal's identity to determine the resources a principal can access. See also: authentication, principal. |
| **Blittable** | Blittable is the new term for isomorphic. |
| **Boxing** | The conversion of a value type instance to an object, which implies that the instance will carry full type information at run time and will be allocated in the heap. The Microsoft intermediate language (MSIL) instruction set's **box** instruction converts a value type to an object by making a copy of the value type and embedding it in a newly allocated object. See also: unboxing, value type. |
| **Building block services** | The building block services are a user-centric set of XML Web services which move control of user data from applications to users. They enable personalized simplicity and consistency across applications, services, and devices, while ensuring user consent is the basis for all transactions. Building block services include Passport (for user identification) and services for message delivery, file storage, user-preference management, calendar management, and other functions. Microsoft will offer a few building block services in areas that are critical to the infrastructure of .NET, and a wide range of partners and developers will significantly expand the set of building block services. |
| **CCW** | See definition for COM callable wrapper. |
| **CLS** | See definition for Common Language Specification. |
| **CLS-compliant** | Code that publicly exposes only language features that is in the Common Language Specification. CLS compliance can apply to classes, interfaces, components, and tools. See also: Common Language Specification (CLS). |
| **Code access security** | A mechanism provided by the runtime whereby managed code is granted permissions by security |

©2001 Microsoft Corporation

| | |
|---|---|
| | policy and these permissions are enforced, limiting what operations the code will be allowed to perform. To prevent unintended code paths from exposing security vulnerability, all callers on the call stack must be granted the necessary permissions (possibly subject to override by assertion or denial). |
| **Code-behind class** | A class that is accessed by an .aspx file, but resides in a separate file (such as a .dll or .cs file). For example, you can write a code-behind class that creates an ASP.NET custom server control, contains code that is called from an .aspx file, but does not reside within the .aspx file. |
| **COM callable wrapper (CCW)** | A proxy object generated by the runtime so that existing COM applications can use managed classes, including .NET Framework classes, transparently. |
| **Channels** | Channels are objects that transport messages between applications across remoting boundaries, whether between application domains, processes, or computers. A channel can listen on an endpoint for inbound messages, send outbound messages to another endpoint, or both. This enables you to plug in a wide range of protocols, even if the common language runtime is not at the other end of the channel. |
| **CLR** | The Common Language Runtime is the execution engine for .NET Framework applications.<br>It provides a number of services, including the following:<br>• Code management (loading and execution)<br>• Application memory isolation<br>• Verification of type safety<br>• Conversion of IL to native code<br>• Access to metadata (enhanced type information)<br>• Managing memory for managed objects<br>• Enforcement of code access security<br>• Exception handling, including cross-language exceptions<br>• Interoperation between managed code, COM objects, and pre-existing DLLs (unmanaged code and data)<br>• Automation of object layout<br>• Support for developer services (profiling, |


©2001 Microsoft Corporation

debugging, and so on)

The following table shows the mapping of all the XML Schema Definition language (XSD) data types to Common Language Runtime (CLR) data types. This mapping is used to build relational schema from an XSD document.

| XSD Type | CLR Type |
|---|---|
| anyURI | System.Uri |
| Base64Binary | Byte[] |
| boolean | bool |
| byte | SByte |
| date | DateTime |
| dateTime | DateTime |
| decimal | decimal |
| double | double |
| duration | TimeSpan |
| ENTITIES | string |
| ENTITY | string |
| float | single |
| gDay | DateTime |
| gMonth | DateTime |
| gMonthDay | DateTime |
| gYear | DateTime |
| gYearMonth | DateTime |
| hexBinary | Byte[] |
| ID | string |
| IDREF | string |
| IDREFS | string |
| int | Int32 |
| integer | Int64 |
| language | string |
| long | Int64 |
| Name | string |
| NCName | string |
| negativeInteger | Int64 |
| NMTOKEN | string |

©2001 Microsoft Corporation

| | | |
|---|---|---|
| | **NMTOKENS** | **string** |
| | **nonNegativeInteger** | **UInt64** |
| | **nonPositiveInteger** | **Int64** |
| | **normalizedString** | **string** |
| | **NOTATION** | **string** |
| | **positiveInteger** | **UInt64** |
| | **QName** | **string** |
| | **short** | **Int16** |
| | **string** | **string** |
| | **time** | **DateTime** |
| | **unsignedByte** | **Byte** |
| | **unsignedLong** | **UInt64** |
| | **unsignedShort** | **UInt16** |
| | **unsingedInt** | **UInt32** |
| **Common language runtime** | The engine at the core of managed code execution. The runtime supplies managed code with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support. | |
| **Common Language Specification (CLS)** | A subset of language features that is supported by a broad set of compliant tools. CLS-compliant components and tools are guaranteed to interoperate with other CLS-compliant components and tools. | |
| **Common object file format (COFF)** | A format in 32-bit programming for executable (image) and object files that is portable across platforms. The Microsoft implementation is called portable executable (PE) file format. See also: portable executable file (PE file). | |
| **Common type system** | The specification that determines how the runtime defines, uses, and manages types. | |
| **Container** | For ASP.NET, see definition for naming container | |
| **Context property** | The implicit state and code to manage that state held on behalf of an object instance. For example, the transaction context property holds the transaction identifier of the transaction that the object is participating in. | |
| **Contract** | The behavior and state that a class provides, which is matched with what a client of that class can | |


©2001 Microsoft Corporation

| | |
|---|---|
| | expect to hold. A contract is expressed partly by the signatures for all public fields, methods, properties, and events of that class. This is augmented by a description (usually in simple descriptive text) of what each field or property represents, together with what each method does. |
| **Custom attribute (Attribute)** | A class used to represent custom metadata. |
| **Custom control** | A control authored by a user or a third-party software vendor that does not belong to the .NET Framework class library. This is a generic term that includes user controls. A custom server control is used in Web Forms (ASP.NET pages). A custom client control is used in Windows Forms applications. |
| **Declarative security check** | Declarative information in metadata for security. Using such declarations, which are usually written as custom attributes, developers can invoke several kinds of security functionality that can be specified: require permissions to bind reference to the code, require permissions to derive a type, demand callers have certain permissions, and so on. See also: imperative security check. |
| **Delegates** | The runtime supports objects called delegates that serve a similar purpose as function pointers in C++. They are type safe, secure, managed objects that always point to a valid object and cannot corrupt the memory of another object. Unlike function pointers in C++, delegates can reference static, virtual, and instance methods. In addition to their use as the equivalent of function pointers, delegates are also used for event handling and callbacks in the .NET Framework. |
| **Denial** | By denying a permission, a method can prevent its callers from exercising the privilege represented by that permission. If a method on the call stack denies permission A, a stack walk checking for permission A will fail unless a valid assertion is found on the stack between the method doing the denial and the method that initiated the check for permission A. |
| **Denial of service attack** | An attack that consumes excessive time, memory, or other resources intending to prevent rightful access to resources. |
| **Deterministic Finalization** | The termination behavior of objects in VB/COM or C++ classes. When the last reference to a COM object is released , the object is immediately terminated. The Class_Terminate procedure is called in VB classes, and all resources that it has |

©2001 Microsoft Corporation

| | |
|---|---|
| | are released. |
| **Disco** | The Web Services Discovery tool discovers the URLs of Web Services located on a Web server and saves documents related to each Web Service on a local disk. |
| **.discomap file** | The .discomap file that is published by a Web Service is an XML document that typically contains links to other resources that describe the Web Service. Web sites that implement a Web Service are not required to support discovery.. Note that you cannot specify a URL to a .discomap discovery document. |
| **.disco** | The Web Services Discovery tool discovers the URLs of Web Services located on a Web server and saves documents related to each Web Service on a local disk in a discovery document (.disco). |
| **Distributed computing** | Distributed computing is a programming model in which processing occurs in many different places (or nodes) around a network. Processing can occur wherever it makes the most sense, whether that is on a server, Web site, personal computer, handheld device, or other smart device.<br>In contrast to the two-node system--the client and the centralized server--prevalent today, Microsoft .NET uses distributed computing. |
| **Domain-neutral** | The practice of loading an assembly without specifying a domain in which the assembly is to run. Loading an assembly in a domain-neutral manner enables the assembly's code and associated run-time data structures to be shared among all application domains within a process. |
| **Download cache** | See definition for assembly cache. |
| **ECMA** | ECMA is an international industry association founded in 1961 and dedicated to the standardization of information and communications systems. |
| **enum** | A special form of value type that inherits from System.Enum and supplies alternate names for the values of an underlying primitive type. An enumeration type has a name, an underlying type, and a set of fields. The underlying type must be one of the built-in signed or unsigned integer types (such as **Byte**, **Int32**, or **UInt64**). The fields are static literal fields, each of which represents a constant. The language you are using assigns a specific value of the underlying type to each field. |
| **Evidence** | The properties of code, such as a digital signature |


©2001 Microsoft Corporation

| | |
|---|---|
| | or the zone or site of its origin, that are used by security policy to grant permissions to the code. |
| **Executable file** | A file in portable executable (PE) file format that can be loaded into memory and executed by the operating system loader. It can be either an .exe or a .dll file. In the .NET context, a PE file must be translated by the common language runtime into native code before it can be executed by the operating system. See also: portable executable (PE) file. |
| **Extensible Markup Language (XML)** | A subset of Standard Generalized Markup Language (SGML) that is optimized for delivery over the Web. XML provides a uniform method for describing and exchanging structured data that is independent of applications or vendors |
| **EZMDI** | Easy Multiple Document Interface |
| **Framework** | See definition for the .NET Framework. |
| **GAC** | See definition for global assembly cache. |
| **Garbage collection (GC)** | The process of transitively tracing through all pointers to actively used objects to locate all objects that can be referenced and then arranging to reuse any heap memory that was not found during this trace. The runtime garbage collector also arranges to compact the memory that is in use to reduce the working space needed for the heap. |
| **Global assembly cache (GAC)** | A machine-wide code cache that stores assemblies specifically installed to be shared by many applications on the computer. Applications deployed in the global assembly cache must have a strong name. See also: assembly cache, strong name. |
| **Globalization** | The process of designing and developing a software product to function in multiple locales. Globalization involves identifying the locales that must be supported, designing features that support those locales, and writing code that functions equally well in any of the supported locales. See also: locale, localization. |
| **Granted permissions** | The permissions determined by security policy that code would be given, allowing it access to resources and giving it identity. The granted permissions are determined by both the requested permissions and what is allowed by the security policy configuration |
| **HailStorm** | As part of the Microsoft® .NET initiative, Microsoft is introducing a user-centric architecture and set of XML Web services, codenamed "HailStorm." |

©2001 Microsoft Corporation

| | |
|---|---|
| | HailStorm will make it easier to integrate the silos of information that exist today. HailStorm services are oriented around people, instead of around a specific device, application, service, or network. They put users in control of their own data and information, protecting personal information and providing a new level of ease of use and personalization. The HailStorm services take advantage of the .NET technologies and architecture that make it possible for applications, devices, and services to work together. These services make user consent the basis for who can access user information, what they can do with it, and how long they have permission. |
| **Heap** | A portion of memory reserved for a program to use for the temporary storage of data structures whose existence or size cannot be determined until the program is running. |
| **HTML server control** | An ASP.NET server control that belongs to the **System.Web.UI.HtmlControls** namespace. An HTML server control maps directly to an HTML element and is declared on an ASP.NET page as an HTML element marked by a **runat=server** attribute. In contrast to Web server controls, HTML server controls do not have an **<asp:***ControlName***>** tag prefix. See also: Web server control. |
| **IDE** | Integrated Development Environment<br><br>Microsoft Visual Studio .NET sports a single shared integrated development environment (IDE) for all the languages within it. It was designed to help developers build their solutions faster with less clutter and with all of the tools easily accessible in any of the languages in the Visual Studio® development system. The Visual Studio .NET IDE has a host of features that bring developers information when they need it and how they want it. |
| **IDL** | Interface Definition Language |
| **IDL Attributes** | Attributes are keywords that specify the characteristics of an interface and of the data and methods within that interface. The set of attributes that applies to a single interface is enclosed in square brackets (**[ ]**). The following topics present the categories of attributes that can be used in an IDL file. |
| **Inheritance** | A child process can inherit several properties and resources from its parent process. |
| **Imperative security check** | A security check that occurs when a security method is called within the code that is being protected. This type of check can be data-driven |


©2001 Microsoft Corporation

| | |
|---|---|
| | and can be isolated to a single location within an object or method. For example, if the name of a file to be protected is known only at run time, then an imperative security check can be invoked by passing the file name as a parameter to a security method. See also: declarative security check. |
| **Intermediate language** | See definition for Microsoft intermediate language. |
| **JIT** | An acronym for "just-in-time," a phrase that describes an action that is taken only when it becomes necessary, such as just-in-time compilation or just-in-time object activation. |
| **JIT compilation** | The compilation that converts Microsoft intermediate language (MSIL) into machine code at the point when the code is required at run time. See also: Microsoft intermediate language. |
| **Lifetime** | The time period that begins when an object is allocated in memory and ends when the garbage collector deletes the object from memory. |
| **Locale** | A collection of rules and data specific to a language and a geographic area. Locales include information on sorting rules, date and time formatting, numeric and monetary conventions, and character classification. See also: localization, globalization. |
| **Localization** | The process of customizing or translating the separated data and resources needed for a specific region or language. See also: globalization. |
| **Managed code** | Code that runs under a "contract of cooperation" with the runtime. Managed code must supply the metadata necessary for the runtime to provide services such as memory management, cross-language integration, code access security, and automatic lifetime control of objects. All code based on Microsoft intermediate language (MSIL) executes as managed code. See also: Microsoft intermediate language (MSIL). |
| **Manifest** | See, Assembly manifest |
| **Metadata** | Information that describes every element managed by the runtime: an assembly, loadable file, type, method, and so on. This can include information required for debugging and garbage collection, as well as security attributes, marshaling data, extended class and member definitions, version binding, and other information required by the runtime. |
| **Method Overloading** | Creating multiple versions of a function that differ |

©2001 Microsoft Corporation

| | |
|---|---|
| | only by their parameter list. |
| **MSI** | Installer Package (.msi) File<br>A Microsoft® Windows® installer package (.msi) file is a storage file containing the instructions and data required to install an application. |
| **Microsoft intermediate language (MSIL)** | A language used as the output of a number of compilers and as the input to a just-in-time (JIT) compiler. The runtime includes several JIT compilers for converting MSIL to native code. See also: JIT compilation. |
| **Module** | A loadable unit, which can contain type declarations and type implementations. The module contains enough information to enable the runtime to locate all implementation bits when the module is loaded. The format for modules is an extension of the Windows portable executable (PE) file format. When deployed, a module is always contained in an assembly. |
| **MSIL** | See definition for Microsoft intermediate language. |
| **Namespace** | A logical naming scheme for grouping related types. For example, the .NET Framework uses a hierarchical naming scheme for grouping types into logical categories of related functionality, such as ASP.NET application framework, or remoting functionality. Design tools can use namespaces to make it easier for developers to browse and reference types in their code. The concept of a namespace is orthogonal to that of an assembly: a single assembly can contain types whose hierarchical names have different namespace roots, and a logical namespace root can span multiple assemblies. In the .NET Framework, a namespace is a logical design-time naming convenience, whereas an assembly establishes the name scope for types at run time. See also: assembly. |
| **Naming container** | Any ASP.NET control, including pages and custom controls, that implements the INamingContainer interface. Such controls create a unique ID namespace for themselves and any child controls they contain. This ensures that any ID attribute given to a control is unique across the entire ASP.NET application in which it is instantiated. |
| **Native code** | Code that has been compiled to processor-specific machine code. |
| **.NET data provider** | A component of ADO.NET that provides access to data from a relational database. |


©2001 Microsoft Corporation

| | |
|---|---|
| **.NET experiences** | .NET experiences are dramatically more personal, integrated end-user computing interactions using connected XML Web services delivered through a new breed of smart devices.<br><br>On a technical level, .NET experiences are a combination of XML Web services and, when appropriate, local application code. |
| **.NET Framework** | A platform for building, deploying, and running Web Services and applications. It provides a highly productive, standards-based, multilanguage environment for integrating existing investments with next generation applications and services, as well as the agility to solve the challenges of deployment and operation of Internet-scale applications. The .NET Framework consists of three main parts: the common language runtime, a hierarchical set of unified class libraries, and a componentized version of ASP called ASP.NET. See also: ASP.NET, common language runtime, .NET Framework class library. |
| **.NET Framework class library** | A CLS-compliant library of classes, interfaces, and value types that are included in the Microsoft .NET Framework SDK. This library provides access to system functionality and is designed to be the foundation on which .NET applications, components, and controls are built. |
| **Orchestration** | Business process orchestration includes the visual modeling, implementation and management of distributed business processes. BizTalk Server 2000 introduces BizTalk Orchestration (BTO), which adds business process orchestration capabilities to the e-commerce architect's toolbox. |
| **PE file** | See definition for portable executable (PE) file. |
| **Permission class** | A class that defines access to a resource or defines an identity by supporting authorization checks. |
| **Permission object** | An instance of a permission class that represents access rights to resources or identity. A permission object can be used to specify a request, demand, or a grant of permission. |
| **Platform invoke** | The functionality provided by the runtime to enable managed code to call unmanaged native DLL entry points. |
| **Polymorphism** | An object-oriented programming term. The ability to have methods with the same name, but different content, for related classes. The procedure to use is determined at run time by the class of the object. For example, related objects might both have Draw |

| | |
|---|---|
| | methods. A procedure, passed such an object as a parameter, can call the Draw method without needing to know what type of object the parameter is. |
| **Portable executable (PE) file** | The file format used for executable programs as well as files to be linked together to form executable programs. |
| **Principal** | In .NET Framework security, represents the identity and role of a user and acts on a user's behalf. See also: role. |
| **Private assembly** | An assembly that is used by only one application. A private assembly is deployed into the directory structure of the application that uses it. See also: shared assembly |
| **Reference type** | A data type that is stored as a reference to the value's location. The value of a reference type is the location of the sequence of bits that represent the type's data. Reference types can be self-describing types, pointer types, or interface types. See also: value type. |
| **Regular expression** | A concise and flexible notation for finding and replacing patterns of text. The notation comprises two basic character types: literal (normal) text characters, which indicate text that must exist in the target string, and metacharacters, which indicate the text that can vary in the target string. You can use regular expressions to quickly parse large amounts of text to find specific character patterns; extract, edit, replace, or delete text substrings, or to add the extracted strings to a collection in order to generate a report. |
| **Requested permissions** | Optionally specified permissions in an assembly that represent the minimum required, optionally desired, and always refused permissions for all code in the assembly. If there is no request, the code is granted the maximum that policy allows. |
| **Resource** | Any non-executable data that is logically deployed with an application. A resource might be displayed in an application as error messages or as part of the user interface. Resources can contain data in a number of forms, including strings, images, and persisted objects. |
| **Role** | A named set of principals that have the same privileges with respect to security. See also: principal. |
| **Round trip** | A sequence of Web Forms stages that begins when a user action requires processing on the server. |


©2001 Microsoft Corporation

| | |
|---|---|
| | When this occurs, a form must be sent (posted) to the server, processed, and returned to the browser. |
| **Runtime** | See definition for common language runtime. |
| **Runtime host** | The environment, such as ASP.NET, Internet Explorer, or the Windows shell, in which the runtime is typically started and managed. Runtime hosts create application domains in which to run managed code on behalf of the user. |
| **Safe mode** | A specific version policy that requests a given assembly be run with the exact version of its dependencies that it was compiled against. |
| **Security hole** | An unintentionally unprotected entry point into an otherwise secure computer, component, application, or other online resource. |
| **Security policy** | The active policy established by the administrator that programmatically generates granted permissions for all managed code based on the code's requested permissions. Code that requires more permissions than policy will grant is not allowed to run. See also: application domain security policy |
| **Server control** | An element that can be dropped into an ASP.NET page or Web Form. Controls are hierarchical, programmable, reusable, server-side objects (with or without a user interface). |
| **Shared assembly** | An assembly that can be referenced by more than one application. An assembly must be explicitly built to be shared by giving it a cryptographically strong name. See also: private assembly, strong name. |
| **Side-by-side execution** | The ability to run multiple versions of the same assembly simultaneously. This can be on the same machine or in the same process or application domain. Allowing assemblies to run side-by-side is essential to support robust versioning in the runtime. |
| **Signature** | The list of types involved in the definition of a method, field, property, or local variable. For a method, the signature includes its name, number of parameters, their types, the type it returns (if any), and its calling convention (default or vararg). The signature for a property is similar to that of a method. The signature for fields and local variables is simply their type (for example, array [0..5] of int). |
| **SOAP** | Simple Object Access Protocol. A simple, XML-based protocol for exchanging structured and type |

| | |
|---|---|
| | information on the Web. The protocol contains no application or transport semantics, which makes it highly modular and extensible. |
| **Stinger** | Software, codenamed Stinger, aimed at setting the standard for multimedia-enabled phones as well as using the strength of its Windows software. |
| **Strong name** | A name that consists of an assembly's identity—its simple text name, version number, and culture information (if provided)—strengthened by a public key and a digital signature generated over the assembly. (Because the assembly manifest contains file hashes for all the files that constitute the assembly implementation, it is sufficient to generate the digital signature over just the one file in the assembly that contains the assembly manifest.) Assemblies with the same strong name are expected to be identical. |
| **Template** | In ASP.NET: A declarative page fragment that is used to provide a visual interface for a templated ASP.NET server control. A template contains presentation elements that include literal text, HTML, data-binding expressions, as well as declarative syntax elements that represent ASP.NET server controls. A template can be persisted as a text file with an .ascx extension. See also: ASP.NET server control, templated control. |
| **Templated control** | An ASP.NET server control that does not itself provide a visual interface but allows its users (page developers) to supply templates that provide a visual interface. When the ASP.NET page parser encounters a templated control, it parses the control's template and dynamically creates child controls that supply the visual interface. The Repeater and DataList ASP.NET server controls are templated controls. The **DataGrid** control is not strictly a templated control, but uses templates to customize its user interface. See also: ASP.NET server control. |
| **Tracing** | A form of debugging, using the Trace class, that tracks and presents execution details of a running application. Tracing is available to any ASP.NET class that implements a **Trace** property. |
| **Type system** | See definition for common type system. |
| **Unboxing** | The conversion of an object instance to a value type. See also: boxing, value type. |
| **UDDI** | The Universal Description, Discovery and Integration (UDDI) project is a sweeping industry initiative. The project creates a platform- |


©2001 Microsoft Corporation

| | | |
|---|---|---|
| | | independent, open framework for describing services, discovering businesses, and integrating business services using the Internet, as well as an operational registry that is available today.<br><br>The UDDI (Universal Description, Discovery and Integration) Business Registry allows businesses to locate information about programmable Web services exposed by their partners. The registry allows any organization to also publish information about the Web services that they expose. As part of the UDDI registry specifications, a programmer's interface was defined that allows someone proficient in using Extensible Markup Language (XML), Simple Open Access Protocol (SOAP), and Hypertext Transport Protocol (HTTP) to interact with the registry directly from a program |
| **Unmanaged code** | | Code that is created without regard for the conventions and requirements of the runtime. Unmanaged code executes in the common language runtime environment with minimal services (for example, no garbage collection, limited debugging, and so on). |
| **User control** | | In ASP.NET: A user-authored server control that enables an ASP.NET page to be re-used as a server control. An ASP.NET user control is authored declaratively and persisted as a text file with an .ascx extension. The ASP.NET page framework compiles a user control on the fly to a class that derives from **System.Web.UI.UserControl**.<br><br>In Windows Forms: A user-authored client-side control that derives from **System.WinForms.UserControl** and is developed by combining existing controls. |
| **Validation server controls** | | A set of server controls included with ASP.NET that test user input in HTML and Web server controls for programmer-defined requirements. Validation controls perform input checking in server code. If the user is working with a browser that supports DHTML, the validation controls can also perform validation using client script |
| **Validator** | | See definition for validation server controls. |
| **Value type** | | A data type that fully describes a value by specifying the sequence of bits that constitutes the value's representation. Type information for a value type instance is not stored with the instance at run time, but it is available in metadata. Value type instances can be treated as objects using boxing. See also: boxing, unboxing, metadata, reference |

| | |
|---|---|
| | type. |
| **Verification** | A part of the compilation process in which code is checked for conformance to a specific set of rules defined to allow proof of certain security requirements. The runtime can verify Microsoft intermediate language (MSIL). See also: Microsoft intermediate language (MSIL). |
| **Version policy** | The rules that specify which version of dependent assemblies to bind to. Version polices are expressed using configuration files. |
| **View state** | The current property settings of an ASP.NET page and those of any ASP.NET server controls contained within the page. ASP.NET can detect when a form is requested for the first time versus when the form is posted (sent to the server), which allows you to program accordingly. |
| **W3C** | The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential as a forum for information, commerce, communication, and collective understanding |
| **Web application** | A set of clients and servers that cooperate to provide the solution to a problem. |
| **Web Forms** | The ASP.NET page framework, which supports server-side controls that render HTML user interface on Web browsers. |
| **Web server control** | An ASP.NET server control that belongs to the **System.Web.UI.WebControls** namespace. Web server controls are richer and more abstract than HTML server controls. A Web server control has an **<asp:***ControlName***>** prefix on an ASP.NET page. See also: ASP.NET server control, HTML server control |
| **Windows Forms** | A rich Windows client library. |
| **Windows XP** | Windows XP is the new Windows desktop operating system for home and business computing. Windows XP will deliver an intelligent, connected, and dependable engine to power .NET experiences, based on a proven technical foundation. |
| **.wsdl** | A .wsdl file is an XML document written in an XML grammar called Web Service description language (WSDL). This file defines how a Web Service behaves and instructs clients as to how to interact with the service. |
| **wsdl.exe** | Web Services Description Language Tool (Wsdl.exe) |


©2001 Microsoft Corporation

| | |
|---|---|
| | The Web Services Description Language tool generates code for ASP.NET Web Services and Web Service clients from WSDL contract files, XSD schemas, and .discomap discovery documents |
| **XML** | XML is the acronym for e**X**tensible **M**arkup **L**anguage, the universal format for structured documents and data on the Web. XML is an industry-standard protocol administered by the World Wide Web Consortium. XML is a key enabling technology for Microsoft .NET.<br><br>See definition for Extensible Markup Language (XML). |
| **XmlDocument** | This class implements the W3C Document Object Model (DOM) Level 1 Core and the Core DOM Level 2. The DOM is and in-memory (cache) tree representation of an XML document and enables the navigation and editing of this document. Because XmlDocument implements the IXPathNavigable interface it can also be used as the source document for the XslTransform class.<br><br>The System.Xml.XmlDataDocument class extends XmlDocument and allows structured data to be stored, retrieved, and manipulated through a relational System.Data.DataSet. This class allows components to mix XML and relational views of the underlying data. |
| **XML Web services** | XML Web services are units of application logic providing data and services to other applications. Applications access XML Web services via standard Web protocols and data formats such as HTTP, XML, and SOAP, independent of how each XML Web service is implemented. XML Web services combine the best aspects of component-based development and the Web, and are a cornerstone of the Microsoft .NET programming model. |
| **XPATH** | XML Path Language |
| **.xsd** | Web Services Description Language XSD schema file (.xsd) |
| **XSL** | In XML, you use style sheets such as Extensible Stylesheet Language (XSL) and cascading style sheets (CSS) to present the data in a browser. XML separates the data from the presentation and the process, enabling you to display and process the data as you wish by applying different style sheets and applications. |
| **XSLT** | XSL Transformations |

| | |
|---|---|
| **Whistler** | Whistler was the code name for Windows XP. See Windows XP. |
| **Windows XP** | Windows XP is the new Windows desktop operating system for home and business computing. Windows XP will deliver an intelligent, connected, and dependable engine to power .NET experiences, based on a proven technical foundation. |
| **WSDL** | Web Services Description Language (WSDL) |
| **WSML** | Web Services Meta Language (WSML) |
| **ZAW** | Zero Administration for Windows (ZAW) |
| **Zero Administration for Windows, ZAW** | IT organizations have spent a great deal of effort in building information to help manage networks, servers, and client platforms. Any new components introduced into the overall management scheme must protect and preserve the existing investments in management information. New components must help IT managers manage and integrate a wide assortment of heterogeneous management components, managed platforms, and managed systems from a central point or location. Microsoft has concentrated on using these customer goals as the basis for enhancing corporate platform manageability and ease of use, and in its overall efforts to help reduce the total cost of ownership under the Zero Administration for Windows® (ZAW) initiative. |


©2001 Microsoft Corporation